

---

# **deluge Documentation**

*Release 2.0b2.dev15*

**Deluge Team**

**July 05, 2017**



---

## Contents

---

<b>1</b>	<b>The Deluge Core</b>	<b>3</b>
1.1	Deluge RPC . . . . .	3
<b>2</b>	<b>Deluge's Interfaces</b>	<b>11</b>
2.1	Deluge GTK UI . . . . .	11
2.2	Deluge Web UI . . . . .	11
2.3	Deluge Console UI . . . . .	11
<b>3</b>	<b>Indices and tables</b>	<b>13</b>
<b>4</b>	<b>Modules</b>	<b>15</b>



Contents:



## Deluge RPC

### Message Formats

DelugeRPC is a protocol used for daemon/client communication. There are four types of messages involved in the protocol: RPC Request, RPC Response, RPC Error and Event. All messages are zlib compressed rencoded strings and their data formats are detailed below.

#### RPC Request

This message is created and sent by the client to the server requesting that a remote method be called. Multiple requests can be bundled in a list.

**[[request\_id, method, [args], {kwargs}], ...]**

**request\_id (int)** An integer determined by the client that is used in replies from the server. This is used to ensure the client knows which request the data is in response to. Another alternative would be to respond in the same order the requests come in, but this could cause lag if an earlier request takes longer to process.

**method (str)** The name of the remote method to call. This name can be in dotted format to call other objects or plugins methods.

**args (list)** The arguments to call the method with.

**kwargs (dict)** The keyword arguments to call the method with.

#### RPC Response

This message is created and sent in response to a RPC Request from a client. It will hold the return value of the requested method call. In the case of an error, a RPC Error message will be sent instead.

**[message\_type, request\_id, [return\_value]]**

**message\_type (int)** This will be a RPC\_RESPONSE type id. This is used on the client side to determine what kind of message is being received from the daemon.

**request\_id (int)** The request\_id is the same as the one sent by the client in the initial request. It used on the client side to determine what message this is in response to.

**return\_value (list)** The return value of the method call.

## RPC Error

This message is created in response to an error generated while processing a RPC Request and will serve as a replacement for a RPC Response message.

[**message\_type, request\_id, exception\_type, exception\_msg, traceback**]

**message\_type (int)** This will be a RPC\_ERROR type id.

**request\_id (int)** The request\_id is the same as the one sent by the client in the initial request.

**exception\_type (str)** The type of exception raised.

**exception\_msg (str)** The message as to why the exception was raised.

**traceback (str)** The traceback of the generated exception.

## Event

This message is created by the daemon and sent to the clients without being in response to a RPC Request. Events are generally sent for changes in the daemon's state that the clients need to be made aware of.

[**message\_type, event\_name, data**]

**message\_type (int)** This will be a RPC\_EVENT type id.

**event\_name (str)** This is the name of the event being emitted by the daemon.

**data (list)** Additional data to be sent with the event. This is dependent upon the event being emitted.

## Remote API

```
class deluge.__rpcapi.RpcApi
```

### class core

Methods available in core

**add\_torrent\_file** (*filename, filedump, options*)

**RPC Exported Function** (*Auth Level: 5*)

Adds a torrent file to the session.

**Args:** filename (str): The filename of the torrent. filedump (str): A base64 encoded string of the torrent file contents. options (dict): The options to apply to the torrent upon adding.

**Returns:** str: The torrent\_id or None.

**add\_torrent\_files** (*torrent\_files*)

**RPC Exported Function** (*Auth Level: 5*)

Adds multiple torrent files to the session.

**Args:** torrent\_files (list of tuples): Torrent files as tuple of (filename, filedump, options).

**Returns:** Deferred



**add\_torrent\_magnet** (*uri, options*)

RPC Exported Function (Auth Level: 5)

Adds a torrent from a magnet link.

**Parameters**

- **uri** (*string*) – the magnet link
- **options** (*dict*) – the options to apply to the torrent on add

**Returns** the torrent\_id

**Return type** string

**add\_torrent\_url** (*url, options, headers=None*)

RPC Exported Function (Auth Level: 5)

Adds a torrent from a url. Deluge will attempt to fetch the torrent from url prior to adding it to the session.

**Parameters**

- **url** (*string*) – the url pointing to the torrent file
- **options** (*dict*) – the options to apply to the torrent on add
- **headers** (*dict*) – any optional headers to send

**Returns** a Deferred which returns the torrent\_id as a str or None

**connect\_peer** (*torrent\_id, ip, port*)

RPC Exported Function (Auth Level: 5)

**create\_account** (*username, password, authlevel*)

RPC Exported Function (Auth Level: 10)

**create\_torrent** (*path, tracker, piece\_length, comment, target, webseeds, private, created\_by, trackers, add\_to\_session*)

RPC Exported Function (Auth Level: 5)

**disable\_plugin** (*plugin*)

RPC Exported Function (Auth Level: 5)

**enable\_plugin** (*plugin*)

RPC Exported Function (Auth Level: 5)

**force\_reannounce** (*torrent\_ids*)

RPC Exported Function (Auth Level: 5)

**force\_recheck** (*torrent\_ids*)

RPC Exported Function (Auth Level: 5)

Forces a data recheck on torrent\_ids

**get\_auth\_levels\_mappings** ()

RPC Exported Function (Auth Level: 0)

**get\_available\_plugins** ()

RPC Exported Function (Auth Level: 5)

Returns a list of plugins available in the core

**get\_completion\_paths** (*args*)

RPC Exported Function (Auth Level: 5)

Returns the available path completions for the input value.

**get\_config** ()

RPC Exported Function (Auth Level: 5)

Get all the preferences as a dictionary

**get\_config\_value** (*key*)  
RPC Exported Function (*Auth Level: 5*)  
Get the config value for key

**get\_config\_values** (*keys*)  
RPC Exported Function (*Auth Level: 5*)  
Get the config values for the entered keys

**get\_enabled\_plugins** ()  
RPC Exported Function (*Auth Level: 5*)  
Returns a list of enabled plugins in the core

**get\_external\_ip** ()  
RPC Exported Function (*Auth Level: 5*)  
Returns the external ip address recieved from libtorrent.

**get\_filter\_tree** (*show\_zero\_hits=True, hide\_cat=None*)  
RPC Exported Function (*Auth Level: 5*)  
returns {field: [(value,count)] } for use in sidebar(s)

**get\_free\_space** (*path=None*)  
RPC Exported Function (*Auth Level: 5*)  
Returns the number of free bytes at path  
**Parameters** **path** (*string*) – the path to check free space at, if None, use the default download location  
**Returns** the number of free bytes at path  
**Return type** int  
**Raises** **InvalidPathError** – if the path is invalid

**get\_known\_accounts** ()  
RPC Exported Function (*Auth Level: 10*)

**get\_libtorrent\_version** ()  
RPC Exported Function (*Auth Level: 5*)  
Returns the libtorrent version.  
**Returns** the version  
**Return type** string

**get\_listen\_port** ()  
RPC Exported Function (*Auth Level: 5*)  
Returns the active listen port

**get\_path\_size** (*path*)  
RPC Exported Function (*Auth Level: 5*)  
**Returns the size of the file or folder ‘path’ and -1 if the path is** unaccessible (non-existent or insufficient privs)

**get\_proxy** ()  
RPC Exported Function (*Auth Level: 5*)  
Returns the proxy settings  
**Returns:** dict: Contains proxy settings.  
**Notes:**  
**Proxy type names:** 0: None, 1: Socks4, 2: Socks5, 3: Socks5 w Auth, 4: HTTP, 5: HTTP w Auth, 6: I2P

**get\_session\_state()**  
**RPC Exported Function** (*Auth Level: 5*)  
 Returns a list of torrent\_ids in the session.

**get\_session\_status(keys)**  
**RPC Exported Function** (*Auth Level: 5*)  
**Gets the session status values for ‘keys’, these keys are taking** from libtorrent’s session status.  
 See: <http://www.rasterbar.com/products/libtorrent/manual.html#status>  
**param keys** the keys for which we want values  
**type keys** list  
**returns** a dictionary of {key: value, ...}  
**rtype** dict

**get\_torrent\_status(torrent\_id, keys, diff=False)**  
**RPC Exported Function** (*Auth Level: 5*)

**get\_torrents\_status(filter\_dict, keys, diff=False)**  
**RPC Exported Function** (*Auth Level: 5*)  
 returns all torrents , optionally filtered by filter\_dict.

**glob(path)**  
**RPC Exported Function** (*Auth Level: 5*)

**move\_storage(torrent\_ids, dest)**  
**RPC Exported Function** (*Auth Level: 5*)

**pause\_session()**  
**RPC Exported Function** (*Auth Level: 5*)  
 Pause all torrents in the session

**pause\_torrent(torrent\_ids)**  
**RPC Exported Function** (*Auth Level: 5*)

**queue\_bottom(torrent\_ids)**  
**RPC Exported Function** (*Auth Level: 5*)

**queue\_down(torrent\_ids)**  
**RPC Exported Function** (*Auth Level: 5*)

**queue\_top(torrent\_ids)**  
**RPC Exported Function** (*Auth Level: 5*)

**queue\_up(torrent\_ids)**  
**RPC Exported Function** (*Auth Level: 5*)

**remove\_account(username)**  
**RPC Exported Function** (*Auth Level: 10*)

**remove\_torrent(torrent\_id, remove\_data)**  
**RPC Exported Function** (*Auth Level: 5*)  
 Removes a single torrent from the session.  
**Args:** torrent\_id (str): The torrent ID to remove. remove\_data (bool): If True, also remove the downloaded data.  
**Returns:** bool: True if removed successfully.  
**Raises:** InvalidTorrentError: If the torrent ID does not exist in the session.

**remove\_torrents(torrent\_ids, remove\_data)**  
**RPC Exported Function** (*Auth Level: 5*)

Remove multiple torrents from the session.

**Args:** `torrent_ids` (list): The torrent IDs to remove. `remove_data` (bool): If True, also remove the downloaded data.

**Returns:**

**list:** An empty list if no errors occurred otherwise the list contains tuples of strings, a torrent ID and an error message. For example:

```
[(<torrent_id>, 'Error removing torrent')]
```

**rename\_files** (*torrent\_id*, *filenames*)

**RPC Exported Function** (*Auth Level: 5*)

Rename files in `torrent_id`. Since this is an asynchronous operation by libtorrent, watch for the `TorrentFileRenamedEvent` to know when the files have been renamed.

**Parameters**

- **torrent\_id** (*string*) – the `torrent_id` to rename files
- **filenames** (*((index, filename), ..)*) – a list of index, filename pairs

**Raises** `InvalidTorrentError` – if `torrent_id` is invalid

**rename\_folder** (*torrent\_id*, *folder*, *new\_folder*)

**RPC Exported Function** (*Auth Level: 5*)

Renames the 'folder' to 'new\_folder' in 'torrent\_id'. Watch for the `TorrentFolderRenamedEvent` which is emitted when the folder has been renamed successfully.

**Parameters**

- **torrent\_id** (*string*) – the torrent to rename folder in
- **folder** (*string*) – the folder to rename
- **new\_folder** (*string*) – the new folder name

**Raises** `InvalidTorrentError` – if the `torrent_id` is invalid

**rescan\_plugins** ()

**RPC Exported Function** (*Auth Level: 5*)

Rescans the plugin folders for new plugins

**resume\_session** ()

**RPC Exported Function** (*Auth Level: 5*)

Resume all torrents in the session

**resume\_torrent** (*torrent\_ids*)

**RPC Exported Function** (*Auth Level: 5*)

**set\_config** (*config*)

**RPC Exported Function** (*Auth Level: 5*)

Set the config with values from dictionary

**set\_torrent\_auto\_managed** (*\*args*, *\*\*kwargs*)

**RPC Exported Function** (*Auth Level: 5*)

Deprecated: Use `set_torrent_options` with 'auto\_managed'

**set\_torrent\_file\_priorities** (*\*args*, *\*\*kwargs*)

**RPC Exported Function** (*Auth Level: 5*)

Deprecated: Use `set_torrent_options` with 'file\_priorities'

**set\_torrent\_max\_connections** (*\*args*, *\*\*kwargs*)

**RPC Exported Function** (*Auth Level: 5*)

Deprecated: Use `set_torrent_options` with 'max\_connections'

**set\_torrent\_max\_download\_speed** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'max\_download\_speed'

**set\_torrent\_max\_upload\_slots** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'max\_upload\_slots'

**set\_torrent\_max\_upload\_speed** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'max\_upload\_speed'

**set\_torrent\_move\_completed** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'move\_completed'

**set\_torrent\_move\_completed\_path** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'move\_completed\_path'

**set\_torrent\_options** (*torrent\_ids, options*)  
 RPC Exported Function (Auth Level: 5)  
 Sets the torrent options for torrent\_ids  
**Args:** torrent\_ids (list): A list of torrent\_ids to set the options for. options (dict): A dict of torrent options to set. See torrent.TorrentOptions class for valid keys.

**set\_torrent\_prioritize\_first\_last** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'prioritize\_first\_last'

**set\_torrent\_remove\_at\_ratio** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'remove\_at\_ratio'

**set\_torrent\_stop\_at\_ratio** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'stop\_at\_ratio'

**set\_torrent\_stop\_ratio** (*\*args, \*\*kwargs*)  
 RPC Exported Function (Auth Level: 5)  
 Deprecated: Use set\_torrent\_options with 'stop\_ratio'

**set\_torrent\_trackers** (*torrent\_id, trackers*)  
 RPC Exported Function (Auth Level: 5)  
 Sets a torrents tracker list. trackers will be [{"url", "tier"}]

**test\_listen\_port** ()  
 RPC Exported Function (Auth Level: 5)  
 Checks if the active port is open  
**Returns** True if the port is open, False if not  
**Return type** bool

**update\_account** (*username, password, authlevel*)  
 RPC Exported Function (Auth Level: 10)

**upload\_plugin** (*filename, filedump*)

**RPC Exported Function** (*Auth Level: 5*)

**This method is used to upload new plugins to the daemon. It is used** when connecting to the daemon remotely and installing a new plugin on the client side. 'plugin\_data' is a xmlrpc.Binary object of the file data, ie, plugin\_file.read()

**class** `RpcApi.daemon`

Methods available in daemon

**authorized\_call** (*rpc*)

**RPC Exported Function** (*Auth Level: 1*)

Determines if session auth\_level is authorized to call RPC.

**Args:** rpc (str): A RPC, e.g. core.get\_torrents\_status

**Returns:** bool: True if authorized to call RPC, otherwise False.

**get\_method\_list** ()

**RPC Exported Function** (*Auth Level: 5*)

Returns a list of the exported methods.

**shutdown** (*\*args, \*\*kwargs*)

**RPC Exported Function** (*Auth Level: 5*)

---

## Deluge's Interfaces

---

Interfaces.

### Deluge GTK UI

### Deluge Web UI

The Deluge web interface is intended to be a full featured interface built using the ExtJS framework, running on top of a Twisted webserver.

### SSL Configuration

By default the web interface will use the same private key and certificate as the Deluge daemon. If you wish to use a different certificate/key (see [How to Create a SSL Certificate](#) for information on creating one) you are able to specify which you want to use.

There are 2 ways to enable SSL encryption in the webserver, 1 is to specify it in your configuration (accessible via the Preferences window). The other is to add '-ssl' when running the webserver, which will override the configuration value and enable SSL.

### Deluge Console UI





## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## CHAPTER 4

---

Modules

---



**A**

add\_torrent\_file() (deluge.\_\_rpcapi.RpcApi.core method), 4  
 add\_torrent\_files() (deluge.\_\_rpcapi.RpcApi.core method), 4  
 add\_torrent\_magnet() (deluge.\_\_rpcapi.RpcApi.core method), 4  
 add\_torrent\_url() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 authorized\_call() (deluge.\_\_rpcapi.RpcApi.daemon method), 10

**C**

connect\_peer() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 create\_account() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 create\_torrent() (deluge.\_\_rpcapi.RpcApi.core method), 5

**D**

disable\_plugin() (deluge.\_\_rpcapi.RpcApi.core method), 5

**E**

enable\_plugin() (deluge.\_\_rpcapi.RpcApi.core method), 5

**F**

force\_reannounce() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 force\_recheck() (deluge.\_\_rpcapi.RpcApi.core method), 5

**G**

get\_auth\_levels\_mappings() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 get\_available\_plugins() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 get\_completion\_paths() (deluge.\_\_rpcapi.RpcApi.core method), 5

get\_config() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 get\_config\_value() (deluge.\_\_rpcapi.RpcApi.core method), 5  
 get\_config\_values() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_enabled\_plugins() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_external\_ip() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_filter\_tree() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_free\_space() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_known\_accounts() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_libtorrent\_version() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_listen\_port() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_method\_list() (deluge.\_\_rpcapi.RpcApi.daemon method), 10  
 get\_path\_size() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_proxy() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_session\_state() (deluge.\_\_rpcapi.RpcApi.core method), 6  
 get\_session\_status() (deluge.\_\_rpcapi.RpcApi.core method), 7  
 get\_torrent\_status() (deluge.\_\_rpcapi.RpcApi.core method), 7  
 get\_torrents\_status() (deluge.\_\_rpcapi.RpcApi.core method), 7  
 glob() (deluge.\_\_rpcapi.RpcApi.core method), 7

**M**

move\_storage() (deluge.\_\_rpcapi.RpcApi.core method), 7

**P**

pause\_session() (deluge.\_\_rpcapi.RpcApi.core method), 7

pause\_torrent() (deluge.\_\_rpcapi.RpcApi.core method), 7

## Q

queue\_bottom() (deluge.\_\_rpcapi.RpcApi.core method), 7

queue\_down() (deluge.\_\_rpcapi.RpcApi.core method), 7

queue\_top() (deluge.\_\_rpcapi.RpcApi.core method), 7

queue\_up() (deluge.\_\_rpcapi.RpcApi.core method), 7

## R

remove\_account() (deluge.\_\_rpcapi.RpcApi.core method), 7

remove\_torrent() (deluge.\_\_rpcapi.RpcApi.core method), 7

remove\_torrents() (deluge.\_\_rpcapi.RpcApi.core method), 7

rename\_files() (deluge.\_\_rpcapi.RpcApi.core method), 8

rename\_folder() (deluge.\_\_rpcapi.RpcApi.core method), 8

rescan\_plugins() (deluge.\_\_rpcapi.RpcApi.core method), 8

resume\_session() (deluge.\_\_rpcapi.RpcApi.core method), 8

resume\_torrent() (deluge.\_\_rpcapi.RpcApi.core method), 8

RpcApi (class in deluge.\_\_rpcapi), 4

RpcApi.core (class in deluge.\_\_rpcapi), 4

RpcApi.daemon (class in deluge.\_\_rpcapi), 10

## S

set\_config() (deluge.\_\_rpcapi.RpcApi.core method), 8

set\_torrent\_auto\_managed() (deluge.\_\_rpcapi.RpcApi.core method), 8

set\_torrent\_file\_priorities() (deluge.\_\_rpcapi.RpcApi.core method), 8

set\_torrent\_max\_connections() (deluge.\_\_rpcapi.RpcApi.core method), 8

set\_torrent\_max\_download\_speed() (deluge.\_\_rpcapi.RpcApi.core method), 8

set\_torrent\_max\_upload\_slots() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_max\_upload\_speed() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_move\_completed() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_move\_completed\_path() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_options() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_prioritize\_first\_last() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_remove\_at\_ratio() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_stop\_at\_ratio() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_stop\_ratio() (deluge.\_\_rpcapi.RpcApi.core method), 9

set\_torrent\_trackers() (deluge.\_\_rpcapi.RpcApi.core method), 9

shutdown() (deluge.\_\_rpcapi.RpcApi.daemon method), 10

## T

test\_listen\_port() (deluge.\_\_rpcapi.RpcApi.core method), 9

## U

update\_account() (deluge.\_\_rpcapi.RpcApi.core method), 9

upload\_plugin() (deluge.\_\_rpcapi.RpcApi.core method), 9